# Test Report

## Manual Testing Approach

I employed systematic manual testing to verify functionality across all application features, focusing on real-world usage scenarios and edge cases.

Presentation:https://www.loom.com/share/3e5e686232e4426c8eef556f1d8c0cd6?sid=5e4dad56-cc69-4a81-912d-2797ea999805

## Feature Testing Results

**User Authentication**

- Verified registration form validation and error messages
- Confirmed login with valid/invalid credentials behaves correctly
- Tested protected routes redirect unauthenticated users
- Validated admin-only features are properly restricted

**Channel Management**

- Created channels with various inputs to test validation
- Verified channels display correctly with accurate metadata
- Confirmed only channel creators and admins can delete channels
- Tested channel listing shows proper message counts

**Message & Reply System**

- Posted messages with text, code examples, and image attachments
- Tested nested replies up to 5 levels deep
- Verified upvote/downvote system properly tracks user votes
- Confirmed message deletion removes associated replies

**Search Functionality**

- Tested keyword search across messages and replies
- Verified user search returns accurate results
- Confirmed empty searches and no-result scenarios provide appropriate feedback

## Key Issues Identified & Resolved

1. **State update issue**: Messages weren't appearing immediately after posting without a refresh. Fixed by implementing proper callback handling in the parent component to update the message list.

2. **Image path inconsistency**: Profile pictures appeared inconsistently across components. Created a central avatar URL helper function to standardize references.

3. **Mobile UI problems**: Form elements overlapped on small screens. Improved responsive layout using Bootstrap grid classes.

# Browser Compatibility

Tested successfully on Chrome, Firefox (desktop), Safari (iOS), and Chrome (Android) with consistent functionality across platforms.

# Testing Tools

- Chrome DevTools for debugging and responsive testing
- React Developer Tools for component inspection
- Network monitoring to verify API communication

This testing approach ensured a stable, user-friendly application that meets all project requirements while identifying and addressing potential issues before submission.